Razvoj Web Aplikacija

Predavanje 7

Danas

- Web API – najbolje prakse
- SOAP
- GraphQL
- HATEOAS
- Obrasci
- Web API i MVC

Web API - Najbolje prakse i obrasci

- **RESTful** – naširoko korišten arhitektonski obrazac, de facto standard
  - CRUD: GET, POST, PUT, DELETE
- Ostali arhitekturalni obrasci :
  - **SOAP** – strukturirana razmjena XML-a preko HTTP-a
  - **RPC** – kao RESTful, ali bez uniformnog sučelja (primjeri - /search, /login, /register…)
  - **GraphQL** – ne samo obrazac, već jezik upita (engl. Query language) koji klijentu omogućuje da dobije podatke koje treba
  - **HATEOAS** - Hypermedia as the Engine of Application State, RESTful API koji je obogaćen poveznicama (hipermedija) koji vode klijenta kroz dostupne resurse i akcije

SOAP

- SOAP – Simple Object Access Protocol
- Koristi XML kao jezik za razmjenu podataka i HTTP kao komunikacijski sloj
- Strogo definira format poruke (zahtjev, odgovor)
- Prethodni de facto standard, još uvijek se koristi tamo gdje se očekuje strogi format poruka
- Siguran, zreo, podržava standarde, široko podržan
- Strma krivulja učenja

SOAP zahtjev

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <GetAudioByGenre xmlns="http://www.example.com/ws">
      <GenreId>1</GenreId>
    </GetAudioByGenre>
  </soapenv:Body>
</soapenv:Envelope>
```

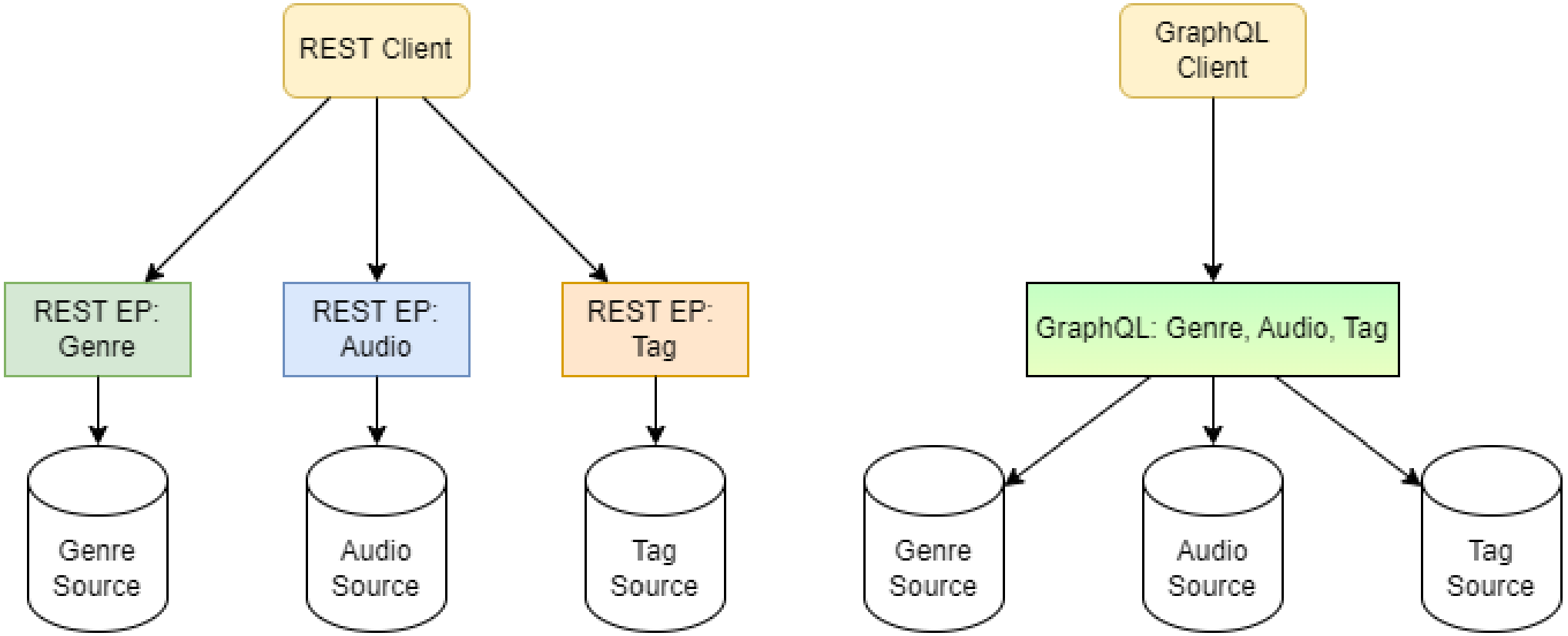
SOAP odgovor

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <GetAudioByGenreResponse xmlns="http://www.example.com/ws">
      <Audios>
        <Audio>
          <Id>1</Id>
          <Title>Song 1</Title>
          <Duration>258</Duration>
          <Genre>
            <Id>1</Id>
            <Name>Rock</Name>
          </Genre>
        </Audio>
        <Audio>
          <Id>2</Id>
          <Title>Song 2</Title>
          <Duration>320</Duration>
          <Genre>
            <Id>1</Id>
            <Name>Pop</Name>
          </Genre>
        </Audio>
      </Audios>
    </GetAudioByGenreResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

GraphQL

- REST implementira operacije niske razine, što može dovesti do brbljavosti u komunikaciji (engl. chattiness)
- GraphQL prikuplja podatke iz mnogih izvora (npr. REST izvori)
- GraphQL klijent ne mora znati iz kojeg su izvora podataka podaci zatraženi
- REST – imperativan (dohvati ili ažuriraj resurs X)
- GraphQL – deklarativan (objasni kako postaviti upit ili mutirati resurs X)
- Oba koriste JSON kao odgovor

GraphQL


REST Client
REST EP: Genre
REST EP: Audio
REST EP: Tag
Genre Source
Audio Source
Tag Source
GraphQL Client
GraphQL: Genre, Audio, Tag
Genre Source
Audio Source
Tag Source

GraphQL primjeri

```
query($genreId: ID!) {
  audios(genreId: $genreId) {
    id
    title
    duration
    genre {
      id
      name
    }
    tags {
      id
      name
    }
  }
}
```

```
mutation {
  updateAudio(id: 123, title: "Updated song", duration: 253) {
    id
    title
    duration
    genre {
      id
      name
    }
    tags {
      id
      name
    }
  }
}
```

HATEOAS primjer 1


```
{ "audios": [
    { "id": 1, "title": "Song 1", "duration": 253,
      "genre": { "id": 1, "name": "Rock" },
      "tags": [
        { "id": 103, "name": "listen" },
        { "id": 204, "name": "alternative" }],
      "_links": {
        "self": { "href": "/api/v1/audios/1" },
        "genre": { "href": "/api/v1/genres/1" },
        "tags": [
          { "href": "/api/v1/tags/103" },
          { "href": "/api/v1/tags/204" }]
      }
    }]
}
```

HATEOAS primjer 2


```
{
  "id": 1,
  "title": "Song 1",
  "duration": "253",
  "genre": ...,
  "tags":  ...,
  "_links": {
    "self": { "href": "/api/v1/audios/1" },
    "edit": { "href": "/api/v1/audios/1", "method": "PUT" },
    "delete": { "href": "/api/v1/audios/1", "method": "DELETE" }
  }
}
```

Ostali tipični Web API obrasci

- CRUD
- Obrazac temeljen na resursima – o jedan resurs je jedan CRUD, savršeno paše u RESTful
- Filtriranje/sortiranje/straničenje
- Verzioniranje
  - GET /api/v1/audios/1 – vraća podatke
  - GET /api/v2/audios/1 – vraća podatke s autorom pjesme i trajanjem u UTC formatu umjesto u sekundama
  - Ne narušava funkcionalnost postojećih klijenata
- Mnoštvo obrazaca za rukovanje pogreškama (otpornost)!

Obrasci otpornosti

- **Timeout** – postavljamo vremensko ograničenje za dovršetak operacije, inače je smatramo neuspjelom
- **Fallback** – ako operacija ne uspije, razmisli o alternativnoj operaciji
- **Retry Pattern** – automatski ponovi neuspjelu operaciju određeni broj puta prije odustajanja; može funkcionirati za neke greške, ali ne i za druge (400 Bad Request); može koristiti *eksponencijalni odmak* (engl. *exponential backoff)* i *jitter* (slučajan pomak vremena)
- **Circuit Breaker** - nakon nekog broja kvarova u vremenskom okviru, prekidač se otvara (nema više zahtjeva); nakon hlađenja, prekidač se napola otvara (ograničen broj ponovnih pokušaja); ako radi dobro, zatvara se (nastavlja rad)
- **Rate Limiting** - ograničavamo broj zahtjeva za klijenta unutar vremenskog okvira

Web API i/ili MVC

- Web API-ji izgrađeni su kao usluge te ih usluge mogu koristiti
- Koriste prvenstveno JSON format (nakada i ne ➔ XML)
- Uglavnom koristite HTTP metode GET, POST, PUT, DELETE
- Kako stvaramo aplikaciju temeljenu na Web API-ju?
  - Stvaramo back-end Web API uslugu koja prihvaća i vraća podatke
  - Stvaramo front-end, uslugu ili program za automatizaciju koji se povezuje s tim back-endom i koristi ga
- MVC nam pomaže predstaviti **korisničko sučelje**, uz logiku aplikacije i razmjenu podataka u istoj aplikaciji

Web API i/ili MVC

- Oboje:
  - Kontroleri i akcije
  - Usluge (engl. services) i međuprogrami (engl. middleware)
  - Prihvaćaju različite formate podataka (Web API uglavnom JSON)
  - Usmjeravanje (različita vrsta usmjeravanja, ali i dalje usmjeravanje)
- Web API
  - Web aplikacija se oslanja na UI klijente kao što su Swagger, Postman, prilagođene mobilne i desktop klijente, usluge, JavaScript kao što su vanilla/jQuery/React/Angular/Vue/Svelte…
- MVC
  - Preglednik je klijent
  - Implementira vlastiti UI (Dotnet Core ➔ Razor, temeljen na HTML-u)

Hvala na pažnji!